The Impact of Contextual Factors on the Security of Code

Dan Shoemaker, Ph.D.
University of Detroit Mercy

Carol Woody, Ph.D.
Carnegie Mellon University
Software Engineering Institute

***Abstract*—Non-technical decisions made in policy, acquisition, governance, resources, processes, and every other aspect of managing software have a direct impact on the resulting operational security. However, these relationships are hidden because the structures we use to govern and organize software do not highlight the security decisions made throughout the life cycle and connect them to the ultimate results. As a result of this obscurity, seemingly appropriate choices result in unacceptable operational security risks because none of the participants recognize the cause and effect linkages.**

The Importance of Contextual Factors

Software security is a critical topic. That is because a single key flaw in a major component can bring down the entire infrastructure. But when we think of software and its security we generally think about it in terms of the software engineering technology, people and processes that are involved in its production and sustainment. We rarely think of the development and maintenance of a system or software artifact within the larger context of how it was acquired, resourced, evolved, or managed. Nonetheless, those larger issues have significant real impacts on the security of every system.

It is reasonable to view software security issues strictly through the lens of the software engineering process. The discrete activities of the development and maintenance phases are well-understood and represent the orthodox understanding of the software industry. They are also the places where flaws are actually created. The activities of global product acquisition, strategic planning, resourcing, and overall business process alignment all take place in the realm of the business outside of the traditional lifecycle. As a result, the relationship between those activities and defects in code are less clear.

Nonetheless, the specific context in which a system is resourced, overseen, managed and assured will have a lot to do with how successfully it performs in actual practice. Software is

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
30 DEC 2014	**N/A**	

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
The Impact of Contextual Factors on the Security of Code	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Mercy) /Carol Woody Dan Shoemaker(U of Detroit	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**12**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

only as good as the people, processes and tools that produce it. The criticality of the development context as introduced by Watts Humphrey in his 1989 publication [1] and further described as a key aspect of the software engineering discipline in his later publication [2] should be considered a fundamental tenet of the software engineering profession. That necessary coordination of context requires a perspective that is more in the realm of business strategy than it is technological. And in that regard, if there are disconnects between the technical processes and the relevant elements of the business operation there is a potential for the injection of serious exploitable vulnerabilities in the actual product.

It is well documented that coherent, well-defined and effective strategic processes are a factor in the production of software and systems. The Software Engineering Institute (SEI) created model after model in the 1990s to underwrite capability [3,4,5] as a critical element of effective software production. Likewise, SEI currently utilizes three different large-scale approaches; two which characterize the capability maturity of the overall process [6,7]; as well as another to describe the strategic maturity of services [8]. That is not to mention the various models of process capability that have been produced by the people at ISO [9,10].

This body of evidence leads us back to the initial premise, which is that the larger context is going to directly impact the correctness, and by implication the security, of every system, or software product. Greater attention has to be paid to contextual factors, which have been largely ignored in the debate about "why Johnny can't code."

Security Impacts of Contextual Factors

Attention has to be given to the impact of business strategy, organizational controls, business process alignment and strategic resourcing decisions on the resulting software and its security.

<u>Business Strategy</u>

The elements of business strategy that impact software and system security include such items as acquisition outsourcing decisions. But they also comprise decisions about organizational development strategy, staffing and training policy.

From a security perspective, the most important choice that an organization is likely to make is the buy-versus-make decision. Many business advantages are associated with commercial-off-the-shelf (COTS) products [11]. Nevertheless, the General Accounting Office lists five areas of high risk in a COTS strategy [12]. Those are: 1) malware, 2) counterfeits, 3) supply chain breakdowns, 4) supplier incapability, and 5) software defects. All of those areas can introduce major security concerns into the organization's electronic infrastructure. Consequently, an intelligent supply chain risk management strategy is an essential component of good system and software security practice.

In addition to supply chains there are the risks associated with the organization's acquisition process itself including specification failures, changing requirements, time and cost over-runs due to lack of control of the process, and insecure product selection [13]. All of these potential issues have to be factored into the outcome of the acquisition process. And very few of these difference-making factors are seen as being a consideration of the acquisition strategy itself. The acquisition strategy chosen to divide system components among several vendors, mandate a reliance on small businesses, rely on commercial off the shelf products (COTS), include open source (or not), purchase from foreign vendors, rely on legacy, and leverage existing hardware infrastructure just to mention a few of the options each contribute to the resulting attack surface, interfaces, and operational security capabilities of the resulting implementation.

These all have real consequences with profound security implications. And thus they are all valid areas of long-term security concern. An organization that allows promiscuous, or unmanaged outsourcing activity is literally playing Russian roulette with the likelihood of system, or software exploitation. Accordingly the points of failure associated with system and software acquisition all have to be thought about and dealt with as part of the overall process of ensuring the integrity and trustworthiness of an organization's electronic infrastructure [14].

The actual technical work also requires a larger perspective. Organizational culture is shaped by strategic policies, which are explicitly defined by the organization in order to ensure the proper level of employee motivation, discipline and training. The Software Engineering Institute issued a model focused entirely on the importance of people to the software development effort [4].

The requisite motivation, discipline and level of skill have to be explicitly fitted to the overall organizational mission and maintained as such. That includes documenting the performance expectations for the work to be done in order to ensure clarity, the promulgation of those expectations to all members of the organization in order to ensure common understanding, and the monitoring of employee performance in order to ensure effectiveness. Anybody who thinks that the culture of the organization doesn't impact the performance of the work has not been keeping up with the literature [15, 19, 20 to cite a few].

The large-scale effort that is required to acquire, implement, and sustain organizational technology requires deliberate and well-coordinated strategic planning and implementation tasks in order to ensure effectiveness. However, both the technical and the business side of the organization often fail to understand the need to actively tailor those tasks to each organizational application. That lack of understanding is unfortunate since, without such a sustainable strategy it is highly unlikely that the employees of the organization will be properly motivated to produce artifacts at the requisite level of effectiveness and security.

Controls

The concept of business management controls might seem far removed from the issue of security flaws in software. But most of the activities that go into the production of a software system have to be overseen and controlled in some formal fashion [16]. Otherwise the organization runs the risk of important decisions being made at the lowest and most inappropriate levels of the organization [3].

Some form of formal governance control is necessary in order to ensure proper and adequate system assurance. Because of the high degree of skill and specialization required, details about software and systems are largely invisible to anybody above the basic technical level. The organization uses governance control mechanisms to influence and direct the way in which code is produced, maintained, and applied. While this has been the case since the beginning of the profession, the increased impact of technology within the functioning of the organization has greatly expanded the importance of these controls.

The problem is that, without specifically designed management controls, which are designed to enforce visibility, the evolution and even use of the system will take place without the direct involvement of most of the organization, specifically all levels of management [15]. The inability to directly oversee technical work forces managers and the organization in general to rely on the capabilities, and even the good will, of employees who have no ability, or reason, to understand the overall strategic goals of the organization, including the desired level of software and system security.

Security controls are built into technical work in the same way that financial controls are built into the accounting process [16]. They must be intentionally designed discrete, systematic behaviors that can measure performance and then move the necessary information to the right decision maker at the right time. Controls allow the organization to both understand, as well as control the present status and long-term evolution of their systems. In the realm of software engineering the primary example of such controls would be the formally planned unit and integration tests and reviews that take place during development. Another example would be all of the formal steps taken to ensure proper configuration management [17].

The development and maintenance of systems as a whole has to be carefully coordinated in order to assure against the types of faults that are the basis for most of the exploits listed in the Common Weakness Enumeration (CWE). However, an effective governance system is also necessary to ensure that corrective action for all of identified defects is systematically initiated, overseen and then signed-off on. Thus, it can be shown that a rationally planned, implemented and executed governance control system is an important aspect of secure code.

Nevertheless the design and implementation of those controls is often left in the hands of business managers, who are no more knowledgeable about the software engineering process than software engineers are about financial accounting. To counteract this separation of knowledge, the organization as a whole has to make a concerted team effort to come up with meaningful and effective controls. This process does not take place by accident. It has to be planned and implemented as part of an overall software security effort. In that respect, control

design and implementation is as much a part of the overall assurance process as static tests [21].

Reliance on incremental development places an even greater burden on these management decisions that directly impact operational security. Who will be making the determination as to when the operational security is sufficient to justify operational deployment and on what basis will they make their decisions? Planning for operational security must be included from the start [29].

Alignment

Proper business process to system alignment is a function of broad scale strategic management [21]. In essence, proper alignment ensures that all software and systems interact optimally with each other and all of the communities of interest that use them. The aim is to produce optimum performance and value for the organization [21, 22].

The aim of strategic alignment is to find the best top-down fit of all of the well-defined lifecycle primary, supporting, ancillary and management processes that are involved in the production and sustainment of software. Alignment is accomplished using explicit process engineering methods best characterized by the ISO 12207-2008 model [26]. This is a very high level and concept based design exercise, with a concrete outcome in the form of a lifecycle infrastructure fitted to the specific environment of that particular organization.

Nevertheless, there are distinct payoffs for proper alignment in the system and software assurance universe. The need to maintain clear and robust linkages between systems ensures against attacks on the interface between systems and users. As a result, strategic alignment becomes a crucial issue in the maintenance of suitable software security. Consideration of alignment on the user interface can also ensure against social engineering scams. Those types of attacks are common methods for exploitation of gaps and misalignments in system operation [23].

The activities that ensure proper alignment are a function of two large software engineering processes, software quality assurance, which in this era also implies security assurance, and

classic configuration management [24]. Those processes are well-defined in the Software Engineering Body of Knowledge (SWEBOK) and can be customized to any application aimed at maintaining monolithic alignment between the systems and software assets of any organization [25]. The ability to align all system and software assets in optimum operational harmony produces real outcomes. Those outcomes include attack surface reduction, gap assurance, and protection against the injection and over-run exploits that comprise the SANS top 25 list [27].

NIST in the recent release of the special publication NIST SP 800-160 Systems Security Engineering [30] defined alignment of security engineering with systems engineering and directly related the tasks of security to the engineering tasks described in ISO/IEC 15288 [31].

The problem comes from the fact that alignment is a strategic activity that can only be enforced at the top of the organization. The primary concern is that this process is rarely carried out in most businesses simply because C-Level executives see system and software alignment as "technical" work. Nonetheless, anything less than total alignment introduces the prospect of security vulnerabilities and is therefore likely to allow breaches that impact the overall mission of the organization. Thus upper level managers have to specifically authorize and delegate the responsibility for alignment in the same manner as the other major functions of the organization.

This is usually in the form of high-level technical managers with the authority to make strategic decisions about system development and deployment across the organization as a whole. It is necessary to focus that authority into a single coordinating entity in order to ensure uniform development of the larger system. It is also important to centralize authority for alignment into a single place for the purposes of oversight and enforcement.

Resourcing

The strategic business processes that most directly impact the security of systems and software are the resourcing decisions. A product is no better than the sum of the people who make it, the tools they utilize and the environment within which the work is performed. Accordingly, it

is in the decisions that provide those resources that risks can be directly weighed and evaluated and eventually either accepted, or mitigated [28].

The primary decision factors in resourcing far predate software and computers. Those are the classic elements of time and money. Decisions like schedule and delivery date impact the time available for assurance as well as the level and degree of inspection and testing. In the larger software engineering sense they also impact the amount of time that can be devoted to getting the specification and design right. Money dictates the number and types of people who can be devoted to a project. Funding impacts the tools available to verify designs and identify and remove defects. It also requires time to learn and apply tools effectively.

Staff capability impacts practically every aspect of the quality and security of the system and software portfolio. Nonetheless, individual staff capability is tied to resource decisions made in the larger sense. Those include such standard resource items as salary and staffing levels, which determines the type and level of talent available to a project. It also includes the general environment and the sophistication of the equipment that is utilized to do the actual work. Poorly staffed and supported software engineering teams are more likely to produce inferior and thus more insecure products.

But resourcing also embraces indirect factors such as whether to outsource. If outsourcing is the approach of choice, then resources determine how rigorously to monitor and control the contractors doing the work. Given the issues discussed earlier that are associated with supply chains, the level and degree of oversight and management of the outsourcing process can determine whether the products that are then integrated into the business operation are secure, or insecure.

Resourcing is often considered in the making of project plans. However it is not clear that resources are ever directly tied to assurance goals in those plans. That is because staff is often described in terms of the roles they fulfill rather than the criticality of the tasks. And the time allotted for project phases is most frequently dictated by the contract with the customer. Therefore it is important to also consider the sensitivity of the tasks themselves when considering how much money to devote to staffing the project. More importantly, it is critical

to factor the assurance case into the business plan. That case is what should be a determinant for software engineering factors such as testing time, test sampling methods and most importantly of all the actual period of time that will be devoted to assurance.

Conclusions

More strategic, non-technical factors can have as great an impact on the security of the code as good programming practice. The decision makers involved in business strategy, organizational controls, business process alignment and strategic resourcing decisions must recognize the impact they have on operational security, understand the importance of coordinating their decisions with technology assurance needs, and accept responsibility for their choices. The strategic decisions affecting the processes, people, and tools should be thought about just as carefully and in as detailed a fashion as the specific software engineering tasks. That is not to suggest that we need to ignore secure coding advice and concentrate solely on strategy, alignment and resourcing. What this suggests is that the problem is a complex of small and large factors, all of which have to be considered as a systematic entity in the assurance of systems and software.

Every one of the factors we discussed has concrete consequences in the real-world and therefore it is impractical to expect secure products without effective planning. Organizational context must be included when considering how to create a secure software engineering production and maintenance environment in order to achieve satisfactory assurance.

Acknowledgment

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

References

1. Humphrey, W., Managing the Software Process, 1989, Addison-Wesley, Reading, MA
2. Humphrey, W., A Discipline for Software Engineering, 1995, Addison-Wesley, Reading, MA
3. Paulk, M. C., Weber, C. V., Curtis, W., Chrissis, M. B., "Capability Maturity Model for Software (Version 1.1)". Technical Report (Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University). CMU/SEI-93-TR-024 ESC-TR-93-177, http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=11955, accessed December 2014
4. Curtis, B., Hefley, W.E., and Miller, S., The People Capability Maturity Model: Guidelines for Improving the Workforce, 2002, Addison Wesley Longman Reading, MA
5. Ahern, D., Armstrong, J., Clouse, A., Ferguson, J., Hayes, W., and Nidiffer, K., CMMI SCAMPI Distilled: Appraisals for Process Improvement. Addison-Wesley Professional, 2005
6. Software Engineering Institute, "CMMI for Development, Version 1.3" CMMI-DEV (Version 1.3), Carnegie Mellon University, 2010
7. Software Engineering Institute, "CMMI for Acquisition, Version 1.3" CMMI-ACQ (Version 1.3), Carnegie Mellon University, 2010
8. Software Engineering Institute, "CMMI for Services, Version 1.3" CMMI-SVC (Version 1.3), Carnegie Mellon University, 2010
9. International Standards Organization, ISO/IEC 15504, Information Technology Process Assessment (parts 2 through 9), ISO/IEC, Geneva, 2011
10. International Standards Organization, ISO/IEC 12207:2008 Systems and Software Engineering — Software Life Cycle Processes, ISO/IEC, Geneva, 2008
11. ResQsoft, “The Benefits and Disadvantages of Commercial Off The Shelf Application,” http://www.resqsoft.com/benefits-disadvantages-commercial-shelf-application.html, accessed November 2014
12. General Accounting Office, “Report to Congressional Committees - GAO High Risk Series, http://www.gao.gov/assets/660/652133.pdf, February 2013, accessed November 2014
13. AcQnotes A Simple Source of DoD Acquisition Knowledge, Software Management - Common Software Failure Causes, http://acqnotes.com/acqnote/careerfields/common-software-failure-causes, accessed November 2014

14. KPMG Consulting, “What went wrong? Unsuccessful Information Technology Projects,” Toronto, Canada, http://p3m.com.au/Reference/KPMG_1997.pdf, accessed November 2014
15. Charette, R. N., “Why Software Fails,” http://spectrum.ieee.org/computing/software/why-software-fails, accessed November 2014
16. Information Systems Audit and Control Association, “Framework for IT Governance and Control,” ISACA, http://www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx, accessed November 2014
17. ANSI/IEEE Std. 1042, IEEE Guide to Software Configuration Management, ANSI/IEEE Std 1042-1987
18. The Open Web Application Security Project, “Command Injection Attacks,” OWASP, https://www.owasp.org/index.php/Command_Injection, accessed November 2014
19. Shahzad, F., Iqbal, Z.,and Gulzar, M., “Impact of Organizational Culture on Employees Job Performance: An Empirical Study of Software Houses in Pakistan,” Journal of Business Studies Quarterly, 2013, Volume 5, Number 2
20. Grondahl, E., and Martinsson, L., “Impact of Organizational Culture on Quality Management,” Chalmers University of Technology, Report No. E2011:025, Gothenburg, Sweden, 2011
21. Van Grembergen, W., and De Haes, S., Enterprise Governance of IT: Achieving Strategic Alignment and Value, Springer, 2009
22. Van Grembergen, W., and De Haes, S., “A Research Journey into Enterprise Governance of IT, Business/IT Alignment and Value Creation”, International Journal of IT/Business Alignment and Governance, Vol. No. 1, 2010, pp. 1–13
23. OWASP, “Top 10 2013 A1: Injection Flaws", OWASP, https://www.owasp.org/index.php/Top_10_2013-A1-Injection, accessed December 2013
24. Meyer, A., “Configuration Management in the Security World,” SANS Corporation, http://www.sans.edu/research/security-laboratory/article/meyer-config-manage, accessed November 2014
25. IEEE Computer Society, “The Software Engineering Body of Knowledge 3.0,” IEEE, http://www.computer.org/portal/web/swebok, accessed November 2014
26. Shoemaker, D. and Sigler, K., Cybersecurity – Engineering a More Secure Software Organization, 014Cengage Publishing, 2013
27. SANS, “CWE/SANS Top 25 Most Dangerous Software Errors,” SANS Corporation, http://www.sans.org/top25-software-errors/, accessed November 2014
28. McGraw, G., The Role of Architectural Risk Analysis in Software Security, Pearson, 2006
29. Woody, C., “Agile Security – Review of Current Research and Pilot Usage” Carnegie Mellon University Software Engineering Institute, April 2013, https://resources.sei.cmu.edu/asset_files/whitepaper/2013_019_001_70236.pdf, accessed November 2014
30. National Institute of Standards and Technology, “DRAFT Systems Security Engineering: An Integrated Approach to Building Trustworthy Resilient System,” NIST SP 800-160, May 12 2014, http://csrc.nist.gov/publications/drafts/800-160/sp800_160_draft.pdf accessed November 2014
31. International Standards Organization, "Systems and software engineering – System life cycle processes", ISO/IEC 15288:2008, ISO/IEC, Geneva, 2008